Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего образования

«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

Лабораторная работа №2  
по дисциплине «Параллельное программирование»

**Программирование взаимодействующих потоков**

|  |  |
| --- | --- |
| Факультет: | ПМИ |
| Группа: | ПМИ-91 |
| Вариант: | 8 |
| Студенты: | Гукасян Кирилл  Федяев Виктор |
| Преподаватели: | Городничев Максим Александрович  Щукин Георгий Анатольевич |

Новосибирск

2023

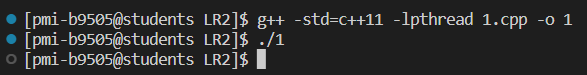
### Цель работы

Познакомиться со средствами планирования и синхронизации потоков; научиться работать с потоками, которые обмениваются информацией между собой.

### Ход работы

### Записать программу из примера 4. Скомпилировать ее и проверить корректность работы

Программа была переписана и скомпилирована корректно.



### Добавить в пример 4 вывод информации о том, какой поток какую задачу взял на исполнение, а также некоторое достаточно продолжительное вычисление в функцию do\_task(). Проверить, что все задачи выполняются, причем каждая – не более одного раза. Отключить вызовы функций pthread\_mutex\_lock() и pthread\_mutex\_unlock() и убедиться, что при отсутствии мьютекса некоторые задачи будут выполняться дважды, тогда как другие не выполнятся ни разу.

Результат работы с синхронизацией представлен на рисунке 1.

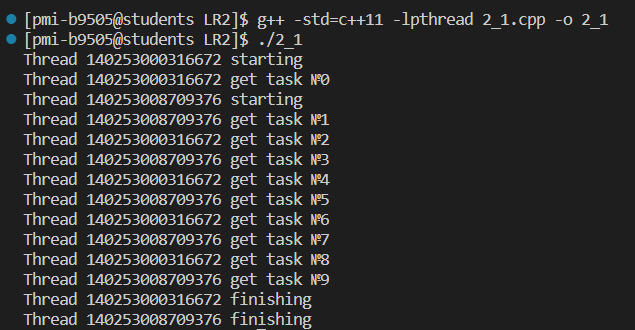


Рисунок 1. Результат с синхронизацией

Результат работы без синхронизации представлен на рисунке 2.

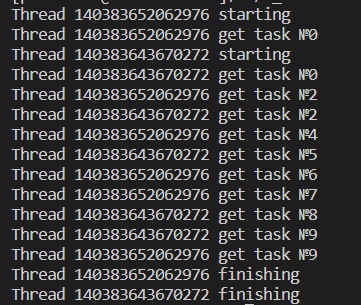
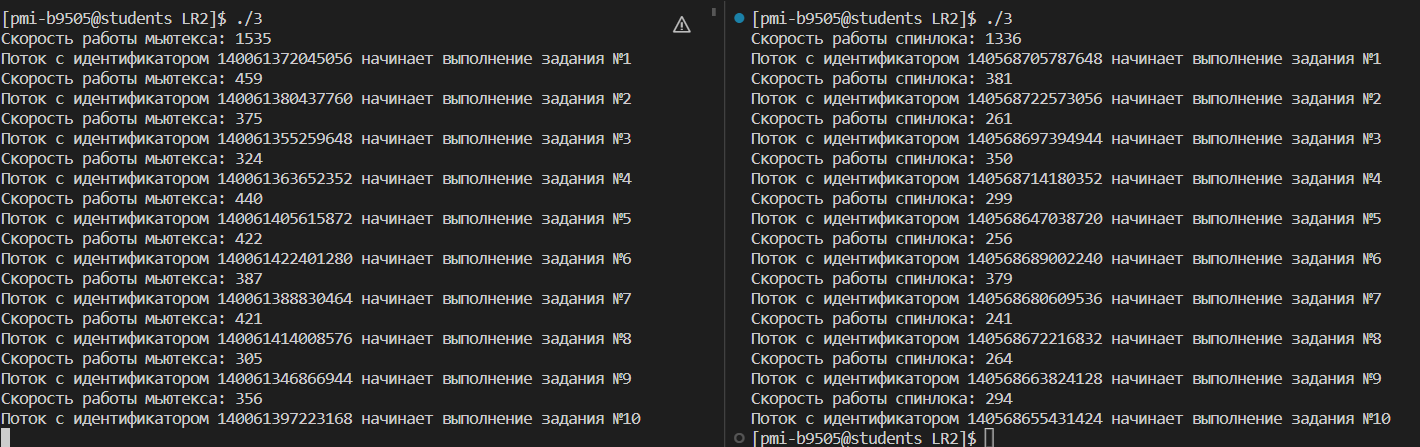


Рисунок 2. Результат без синхронизации

### Сравнить скорости работы примитивов синхронизации – мьютекса и спинлока. Выяснить, как количество потоков, одновременно обращающихся к ресурсу, защищенному примитивом синхронизации, влияет на скорость исполнения программы

Измерение времени работы будем производить от момента непосредственно перед захватом примитива синхронизации ресурса и до момента его освобождения.



Из результатов работы видно, что использование спинлоков позволяет уменьшить время работы программы.

### Реализовать условную переменную с помощью мьютекса и цикла ожидания

Условная переменная была реализована с помощью мьютекса, переменной типа bool и цикла ожидания. Работа условной переменной продемонстрирована на примере потока производителя и потока потребителя (Рисунок 3).

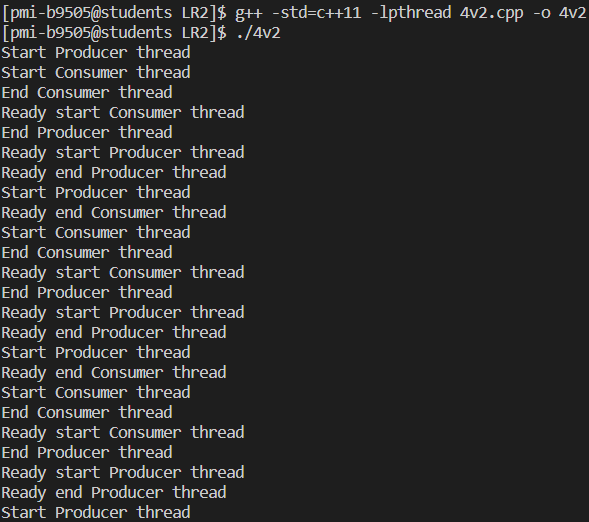


Рисунок 3. Результат с синхронизацией

### Реализовать функцию для выполнения вычислений по модели MapReduce. Подробную информацию о модели MapReduce можно найти в электронном документе: Калита Р. MapReduce5. В качестве параметров функция должна принимать массив данных для обработки, имена функций map и reduce, количество разрешенных потоков. В качестве реализации параллельного применения функции map к каждому элементу массива использовать результаты пункта 8 порядка выполнения лабораторной работы № 1. Реализовать параллельное применение функции reduce к списку, полученному в результате работы функции map

В качестве функции для демонстрации технологии по модели MapeReduce будем использовать функцию, которая находит сумму всех элементов массива. Функция map принимает часть массива и возвращает сумму элементов. Функция reduce суммирует результаты вычисления всех функций map.

Результат работы программы представлен на рисунке 4.

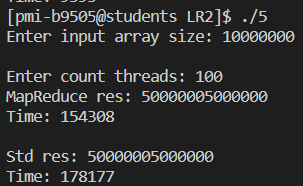


Рисунок 4. Результат работы MapeReduce

Сравнение времени работы для массива размером 1e+7 для разного числа потоков представлено в таблице 2.

### Производитель-потребитель с несколькими потоками-потребителями, одним потоком-производителем и несколькими местами на складе. За основу взять программу из примера 7.

Поток Производитель ожидает сигнала, что склад окажется пустым и пополняет его. Потоки Потребители берут по одному товару со склада и, если склад опустеет, посылают сигнал, что склад опустел. Если склад пустой, то ожидают сигнала о поступлении товаров.

Результат работы для 1 потока Производителя, 3 потоков Потребителей и размером склада равным 4 показан на рисунке 5.

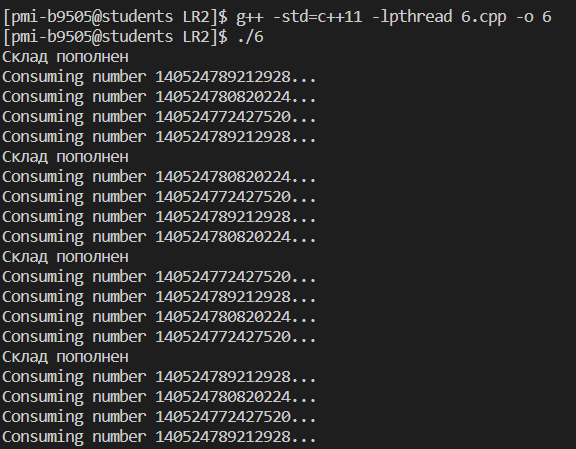


Рисунок 5. Результат программы Производитель-потребитель